WEST

Generate Collection

L10: Entry 1 of 4 File: USPT Oct 9, 2001

DOCUMENT-IDENTIFIER: US 6301582 B1
TITLE: System and method for storage of shared persistent objects

BSPR:
Many common computer systems use an addressing scheme referred to as the two level storage (TLS) model for storing persistent objects. The TLS model stores and manipulates data using two systems: a file manager and a virtual memory system. The virtual memory includes the actual memory and a specialized data file called a swap file. The virtual memory system controls the allocation of address space to different processes. The file manager stores and retrieves data from the permanent storage devices in the form of files.

BSPR:
In particular, in a TLS system persistent data, such as persistent objects, must be stored in files on a disk or other storage medium by the file manager. When a process needs to access a persistent object, the process must contact the file manager which locates the persistent object in a file on backing store and move a copy of the persistent object data into a memory buffer. The persistent object data must then be reconstructed into a persistent object in memory.

BSPR:
Some TLS systems use externalization techniques to store persistent objects. These techniques pull data from the object and write the externalized data to a data file. When the object is needed again, the data must be read in by the file system and the persistent object recreated using state data. This processes requires reading from the file system to a memory buffer, and copying data from the memory buffer to the object. This also creates significant unwanted CPU overhead.

BSPR:
Another addressing scheme is the single level storage (SLS) model. The SLS system maps all of the storage mediums, including persistent storage mediums such as hard drives, into a single address space. This makes the entire data storage system into a single "virtual memory" that is accessed directly using a single, process independent, address space. In an SLS system, persistent data from the permanent storage mediums can be easily copied to real memory using the virtual memory addressing.

DEPR:
Most commodity computer systems today, such as IBM compatible personal computers running IBM's OS/2 or Microsoft's Windows, use a system called two-level store (TLS). TLS systems use a file system for storing data on permanent storage and a virtual memory system for running application processes. Included in the virtual memory system of TLS systems is a specialized data file called a swap file. The swap file is used as "extra memory" to store data for application processes that are too large to be loaded into the limited amount of "real memory".

DEPR:
Thus, no separate steps are required to store persistent objects to backing store, such as those required to externalize object data in TLS systems. Likewise, no separate steps are needed to retrieve persistent objects from backing store. When a persistent object is needed from backing store, the persistent object can be simply copied from backing store into a memory buffer, with no recreation required. Thus, SLS systems eliminate the need to create different runtime and storage versions of persistent objects. Because persistent objects can be simply copied from backing store to/from memory as needed,

processor overhead in dealing with persistent objects is significantly reduced.

DEPR:
It should be understood that in this application the term shared address space refers to the large address space that allows applications to store persistent data using single level store semantics. Likewise, the term native address as used in this application refers to an address that can be used by the underlying system to retrieve data from the page cache. As such, the native address is typically a 32 bit virtual address as used in the two level store of the underlying system. Of course, the underlying system is not required to use 32 bit addressing and can instead use any size native addressing.

DEPR:
To facilitate management of the shared address space 204, a plurality of data structures are provided by the shared persistent virtual storage system 190. These data structures would preferably include cohort data structures for each created cohort, and block data structures for each created block in the cohorts. The cohort data structures would preferably include pointers to block data structures which reside in the corresponding cohort. The block data structures would preferably include the information needed to retrieve data in the corresponding block from data storage 206. For example, in one embodiment each block containing persistent data is stored as a separate file in the file system of data storage 206. In this case, each block data structure would include the name of the file in which the corresponding block is located. When the pager 214 needs to retrieve a page of persistent data from backing store 206 it retrieves the name of the file containing the page from the corresponding block data structure. The shared persistent storage system 190 can then request the required data from the data storage 206 file system using that file name. It should be noted that the preferred embodiment can be adapted to work with any type of data storage system and their associated file systems.

DEPR:
Thus, in the preferred embodiment, when a client process encounters a SAS address, it passes the SAS address to the virtual address translator. The hasher 215 hashes the SAS address and returns a key number n. The key number n is then used to index a hash table to locate a pointer to the page table entry list in the translator lookaside buffer corresponding to that key number. That relatively short list can then be quickly searched for the desired page table entry and the 32-bit address of the data in the page cache. Thus, the hasher 215, the hash table 216 and the translator lookaside buffer 218, are used to quickly locate which page in the page cache, if any, contains the desired information.

CLPR:
7. The apparatus of claim 5 wherein the classloader class creates a persistent class object for encapsulating class data for the persistent object.

CLPR:
32. The method of claim 31 further comprising the step of the persistent classloader object loading class data for the persistent object into a persistent class object in the persistent container object if the class data has not been previously loaded into the persistent container object.

CLPR:
51. The program product of claim 50 wherein the classloader class creates a persistent class object for encapsulating class data for the persistent object.

CCXR:
707/100

CCXR:
707/102

CCXR:
707/8

Print Request Result(s)

Printer Name: cpk2_4y02_gbkeptr
Printer Location: cpk2__4y02

- US006301582: Ok
- US005721918: Ok

OK | Back to List | Logout

WEST

Freeform Search

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: 19 and buffer$

Display: 50 **Documents in Display Format:** FRO **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Image**

Search | Clear | Help | Logout | Interrupt

Main Menu | Show S Numbers | Edit S Numbers | Preferences

Search History

Today's Date: 11/13/2001

DB Name	Query	Hit Count	Set Name
USPT	l9 and buffer$	4	L10
USPT	l8 and (permanent near data)	20	L9
USPT	(persistent near data) and l2	195	L8
USPT	(permanent$ near configuration) and l2	2	L7
USPT	(permanent$ near configuration) and l1	0	L6
USPT	l4 and buffer$	23	L5
USPT	l3 and (data near integrity)	34	L4
USPT	l2 and (permanent$ near data)	139	L3
USPT	((707/$)!.CCLS.)	10899	L2
USPT	(sub near categor$) and corrielus	1	L1

WEST

Generate Collection

L5: Entry 16 of 23 File: USPT Jan 14, 1997

DOCUMENT-IDENTIFIER: US 5594881 A
TITLE: System for updating modified pages of data object represented in concatenated multiple virtual address spaces

BSPR:
Data on storage disks exists in sets of fixed length records accessible in pages. The size of a page varies depending on the system in use. On some systems, a page is 4096 (4K) bytes. Also, on some computers, e.g., virtual machines, the operating system keeps track of disk space in blocks of 256 pages called segments. This is necessary because the hardware requires that a control block be maintained for each segment for virtual address translation. When data is transferred from auxiliary storage to the CPU, pages of data are transferred to a storage buffer in segments.

BSPR:
In order for the CPU to process the data, the data normally should be in main storage. Main storage, however, is limited and is therefore not used to store large amounts of data permanently. On the other hand, vast amounts of data may be stored on data disks. However, accessing data from disks is slow compared to the rate at which it can be processed in main storage. To compensate for the difference in access rates, a data buffer is used. A data buffer is a portion of storage used to hold input and output data temporarily. The data buffer can reside in main storage or expanded storage.

BSPR:
In order to maintain data integrity, the database system takes "checkpoints" of the database at certain intervals to ensure that a consistent version of the database is saved. For example, when a database page is modified, a copy of the page as of the previous checkpoint is kept unchanged, the modified version of the page is copied to disk, and the page directory is updated to point to the new location. Hence, at checkpoint time, the modified version of the database becomes the current copy of the database.

DEPR:
An example of mapping on demand, as provided by one aspect of the present invention, is illustrated in FIG. 14 and is described using the IBM SQL/DS database management system operating on an IBM System/390 computer with the IBM VM/ESA operating system. Before updating a page, the IBM SQL/DS management system begins at step 1404 and copies the requested page to a buffer. It then reissues the MAPMDISK DEFINE request with the PAGEVIEW=ZERO parameter before copying the data back from the local buffer to the data space 608a-608q. Note, PAGEVIEW=ZERO informs the VM/ESA operating system that it should provide an empty real storage page for that data space 608a-608q page rather than reading the contents of the disk 2 when the page is first referenced.

DEPR:
If updating without the use of a local buffer is desired, the page can be remapped with the PAGEVIEW=RETAIN parameter. This first reads the copy of the page from its current disk 2 (old) location before resetting the mapping to the new location. Consequently, the user is presented with the old copy of the page which will be written to the new location.

DEPR:
When the local buffer is used, the remapping is deferred to when the page is moved back from the local buffer to the data space 608a-608q. Performance optimization is implemented by delaying the remapping in an attempt to group

multiple mapping requests into one MAPMDISK DEFINE call.

DEPR:
When a page is moved to the local buffer, step 1404, and a MAPMDISK DEFINE is required, the following is done for a new request:

DEPR:
Like the segment.sub.-- valid bit map 902, a section.sub.-- is.sub.-- modified bit map 1502 is allocated and set to `O` when a data space 608a-608q is referenced for the first time. See step 1208 in FIG. 12. In this case, the section.sub.-- is.sub.-- modified bit map 1502 comprises 1 bit for every 32 pages (or 2048 bytes per data space 608a-608q). The bit representing a set of 32 pages is calculated by dividing the page address by 131,072 (32 pages of 4096 bytes each). Before moving a page from the local buffer back to the data space 608a-608q, the corresponding bit in the section.sub.-- is.sub.-- modified bit map 1502 is checked. See step 1604 in FIG. 16. If it is "OFF" (0), processing proceeds to steps 1606 and 1608 in which the bit is then turned "ON" (1) and a counter is incremented to keep track of the number of bits currently "ON".

DEPR:
With this invention, the database manager I/O function is bypassed. Furthermore, page I/O operations are implemented so that the operating system informs the database manager that a paging I/O operation is needed (and informs it when it is completed), thereby allowing the database manager to perform other tasks in parallel with the I/O operation. This technique results in very significant improvements in database response times without compromising data integrity.

CCXR:
707/200

CCXR:
707/205

WEST

Generate Collection

Search Results - Record(s) 1 through 1 of 1 returned.

☐ 1. Document ID: US 5963948 A

L1: Entry 1 of 1 File: USPT Oct 5, 1999

US-PAT-NO: 5963948
DOCUMENT-IDENTIFIER: US 5963948 A

TITLE: Method for generating a path in an arbitrary physical structure

DATE-ISSUED: October 5, 1999

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Shilcrat; Esther Dina	Stony Brook	NY	11790	

APPL-NO: 8/ 749528
DATE FILED: November 15, 1996

INT-CL: [6] G06F 17/30
US-CL-ISSUED: 707/100; 707/3, 707/4, 707/1, 340/995, 701/200, 701/207, 701/209
US-CL-CURRENT: 707/100; 340/995, 701/200, 701/207, 701/209, 707/1, 707/3, 707/4
FIELD-OF-SEARCH: 364/228, 364/282.1, 364/974.6, 364/242.94, 395/12, 395/200.52, 395/52, 345/326, 380/4, 707/100, 707/4, 707/1, 707/3, 340/995, 701/200, 701/207, 701/209

PRIOR-ART-DISCLOSED:

U.S. PATENT DOCUMENTS

PAT-NO	ISSUE-DATE	PATENTEE-NAME	US-CL
4866661	September 1989	De Prins	364/900
4992940	February 1991	Dworkin	364/401
5127674	July 1992	Lamphere et al.	283/37
5250789	October 1993	Johnsen	235/383
5465291	November 1995	Barrus et al.	379/67
5727129	March 1998	Barrett et al.	395/12

ART-UNIT: 276
PRIMARY-EXAMINER: Kulik; Paul V.
ASSISTANT-EXAMINER: Corrielus; Jean M.
ABSTRACT:

An item locating and path generating computer system creates personalized paths through structures by converting each of a set of items, objects, or locations of particular interest to a user to a corresponding set of destinations within the structure, and creating a path which includes each element in the set of destinations. The generated path may start at a structure entrance and end at a structure exit. The generated path may also reflect user desired characteristics such as least distance, least time, being at a specific point at a specific time, etc. In structures with multiple entrances or exits, multiple paths can be created and evaluated by above user criteria to find the best fit. The personalized paths can be stored, and automatically searched and queried in a variety of ways, such as aggregation. The invention includes a number of methods for creating a set of desired items, objects or locations of particular interest to a user. This

desired items, objects or locations of particular interest to a user. This includes storing elements of this set, along with a desired frequency or another condition (including price). The invention also converts between the units in which an object may be measured (as in, for example, a recipe) and the units in which that object actually occurs in the structure.

15 Claims, 40 Drawing figures

Generate Collection

Term	Documents
SUB.USPT.	178264
SUBS.USPT.	1603
CATEGOR$	0
CATEGOR.USPT.	5
CATEGORATIZED.USPT.	1
CATEGOREIES.USPT.	2
CATEGOREIS.USPT.	2
CATEGOREIS:.USPT.	1
CATEGOREMATICALLY.USPT.	1
CATEGORES.USPT.	4
((SUB NEAR CATEGOR$) AND CORRIELUS).USPT.	1

There are more results than shown above. Click here to view the entire set.

Display 100 Documents, starting with Document: 1

Display Format: FRO Change Format

WEST

Generate Collection

L5: Entry 2 of 23 File: USPT Aug 21, 2001

DOCUMENT-IDENTIFIER: US 6278999 B1
TITLE: Information management system for personal health digitizers

DEPR:
Download Acceptance Software 402 --accepts data for storage in the database, places the received data in a buffer file until the received data can be screened and processed for inclusion in the database

DEPR:
In order to ensure the integrity of the data that is stored in the database 400, the information management system IMS includes a download acceptance process 402 that receives data that is transmitted by a consumer to the information management system IMS and stores the data via path (x) in a temporary file termed "data on hold 409" until the data can be validated. The validation process comprises a review of the format and content of the data to prevent bogus data from corrupting the integrity of the database 400. In particular, the consumer identification information as well as the associated measurement data is screened for data usability and associated demographic information. The proper formatting of the data is verified and then the received data is stored in the data on hold file 409. Once the data stored in this file is reviewed by either information management system IMS personnel and/or further validation software, it is downloaded via path (y) to the permanent data repository of data tables, files and records 408 where it is incorporated into the existing population of data.

CCOR:
707/9

CCXR:
707/3

CCXR:
707/4

CCXR:
707/6